BOTTOM UP PARSING

(ora => sta per $=>_r$)

S::= abABe
A::= Abc | b
B::= d

abbbcde $\in$L(S) ?

Reversed rigthmost

ab<u>b</u>bcde<=ab<u>Abc</u>de<=abA<u>d</u>e<=<u>abABe</u><=S

shift-reduce parsing (LR)

Ricostruire all'indietro una derivazione destra significa saper dire:

Abbbcde

$$a\underline{b}bbcde \;{}_{A::=b}{<=}\; aAbbcde$$

$$ab\underline{b}bcde \;{}_{A::=b}{<=}\; abAbcde$$

$$abb\underline{b}cde \;{}_{A::=b}{<=}\; abAcde$$

$$abbbc\underline{d}e \;{}_{B::=d}{<=}\; abbcBe$$

Quale delle 4 riduzioni e' corretta?,

Quell'unica che fa parte di una derivazione ovvero forma sentenziale destra

Maniglia

Sia $\gamma_1\beta\gamma_2$ una *RFS* allora

A::=β è maniglia

se e solo se

S=>* $\gamma_1 A\gamma_2$ (=> $\gamma_1\beta\gamma_2$ =>….)

Le maniglie ci fanno viaggiare lungo RFS solamente

Shift-Reduce: Un semplice riconoscitore a due operazioni:

lookahead

top

1) *Shift* se non hai superato **la** maniglia
metti il lookahed in pila

2)*Reduce*: altrimenti (lookahed e’ a destra della m.)
estrai dalla pila i simboli di β della maniglia
$A::=\beta$ e rimpiazzali con A

a b b b c d e $

$

all'inizio: a non e' a destra di maniglia in stack
Shift

b b b c d e $

a
$

b non e' a destra di maniglia in stack
Shift

b c d e $

b
b
a
$

b e' a destra di maniglia in stack
Reduce con maniglia A::=b

b c d e $

A
b
a
$

b non e' a destra di maniglia in stack
Shift

c d e $

b
A
b
a
$

c non e' a destra di maniglia in stack
Shift

1) riconoscitore: automa a pila con

driver D a operazioni *shift-reduce*

tabella M con maniglie

2) LR piu' potente di LL:

per individuare la riduzione vediamo l'intera parte sinistra della sentenziale

S::= u S | u A

A::= u A z | u B z

B::= v B | v

MECCANIZIAMO LA SCELTA

Come individuare **le maniglie** scorrendo
la sequenza da sinistra a destra?

Usiamo **viable prefix** (prefisso percorribile)

viable prefix

Sia $\gamma_1\beta\gamma_2$ una qualunque *RFS*

(i.e. $S \Rightarrow^* \gamma_1\beta\gamma_2$)

maniglia di $\gamma_1\beta\gamma_2$ sia $A::=\beta$

(i.e. $S \Rightarrow^* \gamma_1 A\gamma_2 \Rightarrow \gamma_1\beta\gamma_2$)

Allora:

ogni prefisso di $\gamma_1\beta$ è un viable prefix

Insieme dei viable prefix di G

$VP_G = \{prefix(\gamma_1\beta) \mid A::=\beta \in G \; , \; \gamma_1 A \gamma_2 \Rightarrow \gamma_1 \beta \gamma_2 \in RFS$ per qualche $\gamma_2\}$

L'insieme dei viable prefix forma un **automa a stati finito deterministico**

Che ci fornisce la tabella M

se **G e' LR**

TRE differenti automi:

SLR i piú semplici (tabella/FSA piccolo)

LR i piú potenti (tabella/FSA grande)

LALR un compromesso: semplicitá/potenza

L'idea: Gli **Items LR(0)**

Sia $A ::= \alpha\beta$ una produzione della grammatica G
allora: $\boxed{A ::= \alpha.\beta \text{ e' un } \textbf{item LR(0)}}$

$$S \Rightarrow^* \gamma A \delta \Rightarrow \gamma\ \alpha\beta\ \delta$$

α e' un tratto del viable prefix $\gamma\alpha$

$\boxed{A ::= \alpha.\beta \text{ e' un } \textbf{valid item} \text{ per } \gamma\alpha}$

Componiamo valid item per riconoscere viable prefix

La tecnica: Collezione canonica di LR(0) items

raccogliamo in insiemi i *valid item* di *viable prefix* di una stessa *maniglia*.

Su questi insiemi I deve valere la seguente **proprietà** (I=Closure(I))

$\text{Closure}(I) =_{min} I \cup \text{Closure}\{B ::= .\gamma \mid A ::= \alpha.B\beta \in I\}$

B è un noterminale

se $A ::= \alpha.B\beta$ è *valid item* per un prefix

allora anche $B ::= .\gamma$ è *valid* per quello stesso prefix

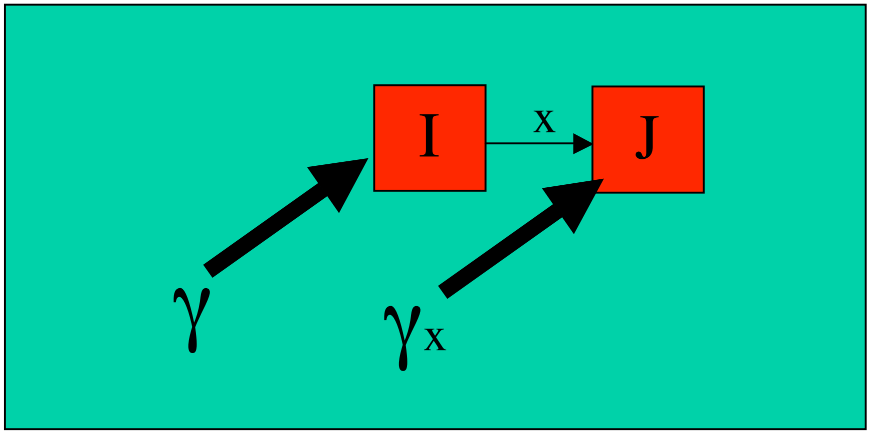
quindi devono stare insieme

$$S \Rightarrow^* \rho A\delta \Rightarrow \rho\, \alpha B\beta\, \delta \Rightarrow^* \rho\, \alpha\gamma\, \delta'$$

Su questi insiemi I,J deve valere la seguente **relazione**

J=Goto(I,x), per qualche insieme I e simbolo x

Goto(I,x)= Closure{$A::=\alpha x.\beta$ | $A::=\alpha .x\beta$ ∈I}
x e' un teminale o nonterminale



Collezione Canonica di LR(0) items: Coll(0)

Formata da un insieme di stati I

* ciascuno chiuso rispetto a Closure
* l’intero insieme chiuso rispetto a Goto

Occorre uno stato iniziale Io:

Aumentiamo la grammatica

$G=\langle S,V,P,s\rangle$ diventa $G'=\langle S\cup\{s'\},V,P\cup\{s'::=S\},s'\rangle$

$Io=Closure(\{s'::=.s\})$

ESEMPIO

S::= aABe
A::= Abc | b
B::= d

$\Rightarrow$

S'::=S
S::= aABe
A::= Abc | b
B::= d

I0:Closure({S'::=.S})={S'::=.S
S::= .aABe}

Calcoliamo gli altri stati J

S’::=S
S::= aABe
A::= Abc | b
B::= d

I0:Closure({S’::=.S}) ={S’::=.S
S::= .aABe}

I1:Goto(I0,S) ={S’::=S.}

I2:Goto(I0,a) ={S::= a.ABe
A::=.Abc
A::=.b}

I3:Goto(I2,A) ={S::= aA.Be
A::=A.bc
B::= .d}

I4:Goto(I2,b) ={A::=b.}

I5:Goto(I3,B) ={S::= aAB.e}

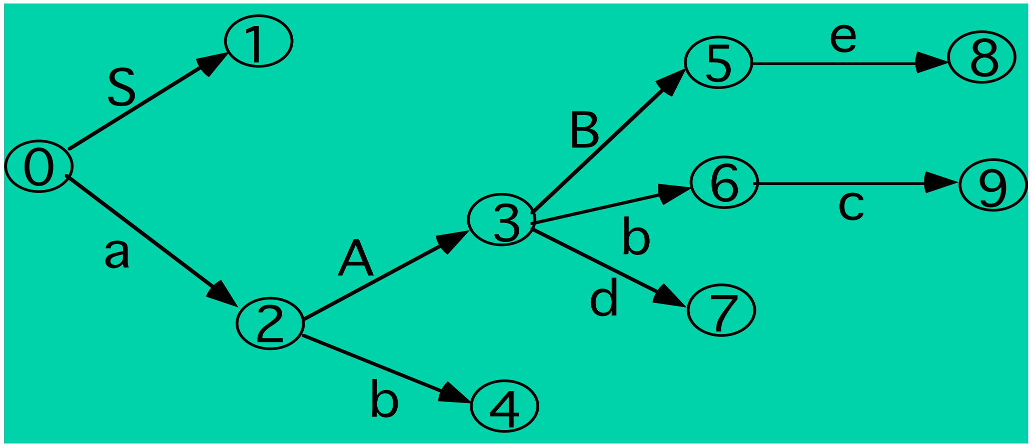
I6:Goto(I3,b) ={A::=Ab.c}

I7:Goto(I3,d) ={B::= d.}

I8:Goto(I5,e) ={S::= aABe.}

I9:Goto(I6,c) ={A::=Abc.}

Ognuno di questi insiemi In contiene
possibili maniglie per prefix che da I0
conducono a In attraverso transizioni goto.



Scoprite la maniglia di
abbcde

Quando non possiamo piu' continuare a percorrere l'automa siamo a destra di una maniglia

TABELLA M per SLR

per ogni stato indica:

action: shift/reduce

goto: nuovo stato dopo reduce

come si costruisce (due sottotabelle):

calcolo della collezione canonica

↓

calcolo dei follows

↓

Calcolo della action

ACTION(i,a)=<shift,j>

se goto(Ii,a)=Ij and $a \in \Sigma$

ACTION(i,a)= <reduce,p>

se $A ::= \alpha . \in Ii$ and $p \equiv A ::= \alpha$ and $a \in follow(A)$

ACTION(i,$)= <accept,->

se $S' ::= S . \in Ii$

GOTO(i,A)=j

se goto(Ii,A)=Ij and A $\in$S(nonterminali)

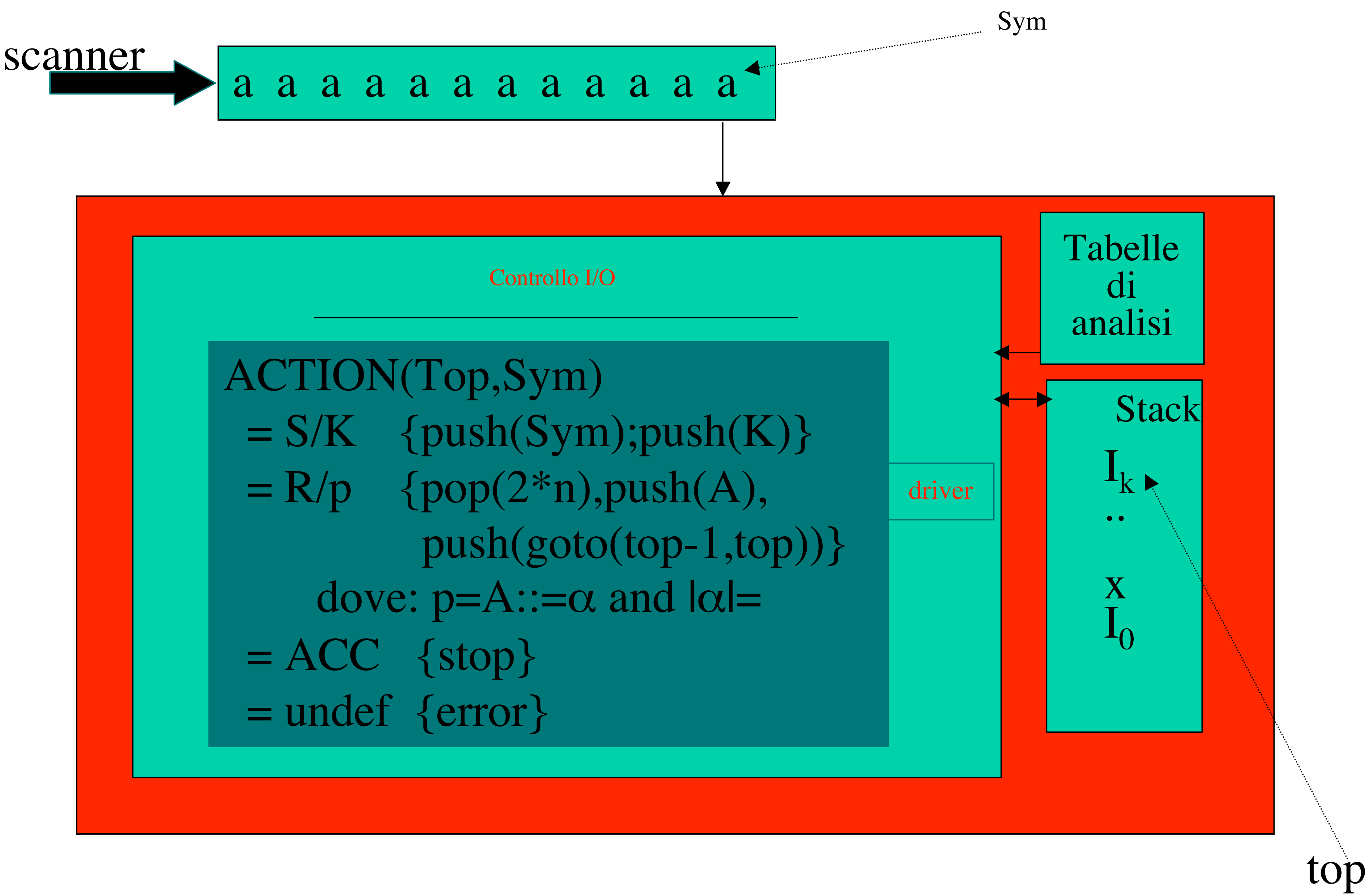
ESEMPIO

	a	b	c	d	e	$
0	S/2					
1						ACC
2		S/4				
3		S/6		S/7		
4		R/2		R/2		
5					S/8	
6			S/9			
7					R/3	
8						R/0
9	..	R/1		R/1		

	S	A	B
0	1		
1			
2		3	
3			5
4			
5			
6			
7			
8			
9	..		

S’::=S
S::= aABe [0]
A::= Abc [1]| b [2]
B::= d [3]

follow(S)={$}
follow(A)={b,d}
follow(B)={e}


scanner
a a a a a a a a a a a a
Sym
Controllo I/O
ACTION(Top,Sym)
= S/K {push(Sym);push(K)}
= R/p {pop(2*n),push(A),
push(goto(top-1,top))}
dove: p=A::=α and |α|=
= ACC {stop}
= undef {error}
driver
Tabelle di analisi
Stack
I_k
..
x
I_0
top

Applicabilita' SLR(1)

proprietà1: (No shift-reduce)

$(\forall\ I_k \in Coll(0)), (\forall\ A ::= \alpha . a\gamma,\ B ::= \beta . \in I_k)$

$a \notin follow(B)$

proprietà2: (No reduce-reduce)

$(\forall\ I_k \in Coll(0)), (\forall\ A ::= \alpha .,\ B ::= \beta . \in I_k)$

$follow(A) \cap follow(B) = \{\}$

Th. Data una grammatica G:

- G∈SLR(1) se e solo se entranbe prop.1 e prop.2
- G ha parser a riduzione 1-simbolo SLR se e solo se entranbe prop.1 e prop.2

SLR
Conclusioni

1. In che senso il riconoscitore bottom-up, basato sui viable prefix e sull'attraversamento della maniglia, non è predittivo ?

2. Se una grammatica G è SLR(1), la tabella:

ACTION può contenere più azioni per una stessa entry ?

GOTO può contenere più stati per una stessa entry ?

3. Cosa significa per una grammatica $G \in SLR(1)$, avere la tabella: ACTION tale che per ogni I, per ogni coppia $a \neq b$,

$$ACTION(I,a)=ACTION(I,b)$$

allorchè $ACTION(I,a) \neq \perp \neq ACTION(I,b)$?

SLR:

Conclusioni (continua)

4. Per ogni grammatica G è decidibile G $\in$SLR(1) ?

5. Per ogni linguaggio **T** è decidibile **T** $\in$SLR(1) ?

6. Perchè, in SLR(1), se $A::=\alpha. \in I_k \in Coll(0)$, allora
$A ::=\alpha$
è maniglia per ogni $a \in follow(A)$?

Nota: **T** $\in$SLR(1) $\equiv \exists$ G $\in$LL(1): L(G) = **T**

SLR: CONFLITTI VERI E FALSI
Analisi bottom-up: LR - LALR

$S \Rightarrow_r^* \gamma A \beta \Rightarrow_r \gamma\alpha\beta$

$S \Rightarrow_r^* \gamma A \beta$ è una sentenziale destra → $First(\beta) \in follow(A)$

$\gamma A \beta \Rightarrow_r \gamma\alpha\beta$ è una deriv. destra → $A ::= \alpha$ è maniglia di $\gamma\alpha\beta$

First(β)∈Follow(A) è una condizione necessaria

Th. (c. necessaria)

$\forall$ G, $\forall \gamma\alpha \in VP_G$

se $A ::= \alpha$ maniglia per $\gamma\alpha\beta$

allora - $First(\beta) \in follow(A)$ &

- $A \to \alpha.$ è valid item

First(β)∈Follow(A)
Condizione Sufficiente o solo necessaria?

$S \Rightarrow_r^* \gamma\alpha\beta$

$\gamma\alpha$ ∈viable prefix
First(β)∈follow(A)
A::=α e' produzione

Possiamo concludere che
A::= α e' maniglia di $\gamma\alpha\beta$?

First(β)∈follow(A) non è condizione sufficiente

Prop. (c. non sufficienza)
∀ G, ∀ $\gamma\alpha \in VP_G$ con valid item A->α.
First(β) ∈ *follow(A)* **non implica** A::=α maniglia di $\gamma\alpha\beta$

Un esempio

S::= A u | a v
A::= a

I0:Closure({S'::=.S}) ={S'::= .S
S::= .Au, S::= .av, A::= .a}
I1:Goto(I0,S) ={S'::= S.}
I2:Goto(I0,A) ={S::= A.u}
I3:Goto(I0,a) ={S::= a.v
A::= a.}
I4:Goto(I2,u) ={S::= Au.}
I5:Goto(I3,v) ={S::=av.}

I0:Closure({S'::=.S}) ={S'::=.S
S::=.Au, S::= .av, S::= .Bv,
A::=.a, B::=.xA}
I1:Goto(I0,S) ={S'::= S.}
I2:Goto(I0,A) ={S::= A.u}
I3:Goto(I0,a) ={S::= a.v
A::= a.}
…. Conflitto: follow(A)={u,v}

S::= A u | a v | B v
A::= a
B::= x A

$S \Rightarrow_r^* a\ v$

A::= a e' produzione
v∈follow(A)

Ma A::=a non e'
Maniglia per av

Solo Necessaria
Una prova

Contro-esempio: Vediamo γ, α, β

S::= uAa | A b | a a
A::= a

$S \Rightarrow_r^* a\ a$
- per $\gamma = \lambda$, $\alpha = a$, $\beta = a$
- $a \in VP$
- A->a. valid item per a

No: infatti non è maniglia - $S \Rightarrow_r^* a\ a$

$LL(1) \not\subset_G SLR(1)$
Ancora un esempio:

S::= uAa | B
B::= Ab | a
A::= ε

$S \Rightarrow_r^* a$

- $\gamma=\lambda$, $\alpha=\lambda$, $\beta=a$
- $\lambda \in VP$
- A->. valid item per λ

No: non è maniglia - $S \Rightarrow_r B \Rightarrow_r a$

RICONOSCIAMO I CONFLITTI VERI
ricordiamo il contesto destro

Ricordiamo il contesto a destra di ogni viable prefix
che ci porta ad una maniglia

I0={S::=.Au, A::= .a...} ----> I3=G(0,a)={..A::=a.}

A::=.a è un item che segue da S::=.Au

solo se (la riduzione ad) A ha come
lookahead il simbolo u

Una nuova classe di Item

$X ::= \alpha . A\beta \in I_k$ ed $A ::= \gamma$ e' una produzione

allora

$A ::= .\gamma \in I_k$ solo per
i simboli nell'insieme $U = First(\beta)$

$First(\beta\ \mathit{look}(X ::= \alpha\ A\beta))$

OCCORRE ACCOPPIARE item LR(0) con insieme U di *simboli di look*ahead

$\boxed{A ::= .\gamma, U} \in I_k$

Item LR(1)

Collezione canonica di LR(1) items: Coll(1)

$S'::=.S,\$ \in I_0$ per definizione

$Clos(I) =_{min} I \cup Clos\{B::= .\gamma,U \mid A::=\alpha.B\beta,V \in Clos(I),$
$U = \{first(\beta x) \mid x \in V\}\}$

B e' un nonterminale
Sia $u \in U$, allora:
se $u \in V$, $\alpha.B\beta$ *propaga* su $B::= .\gamma$
se $u \notin V$, $\alpha.B\beta$ *genera spontaneamente* su $B::= .\gamma$

$Goto(I,x)=Closure\{A::= \alpha x.\beta,U \mid A::=\alpha.x\beta,U \in I\}$

LR(1)

proprietà1: (No shift-reduce)

$(\forall\ I_k \in Coll(1)),\ (\forall\ A::=\alpha.a\gamma,V\ ,\ B::=\beta.,U\ \in I_k)$

$a \notin U$

proprietà2: (No reduce-reduce)

$(\forall\ I_k \in Coll(1)),\ (\forall\ A::=\alpha.,V\ ,\ B::=\beta.,U \in I_k)$

$V \cap U = \{\}$

Th. Data una grammatica G:

- G∈LR(1) se e solo se entranbe prop.1 e prop.2
- G ha parser a riduzione 1-simbolo LR se e solo se entranbe prop.1 e prop.2

Calcolo delle tabelle ACTION e GOTO per LR(1)

ACTION(i,a)=<shift,j>

se goto(Ii,a)=Ij and $a \in \Sigma$

ACTION(i,a)= <reduce,p>

se $A::=\alpha.,U \in Ii$ and

$p \equiv A::= \alpha$ and $a \in U$

ACTION(i,\$)= <accept,->

se $S'::=S.,\$ \in Ii$

GOTO(i,A)=j

se goto(Ii,A)=Ij and $A \in N$

N=insieme dei nonterminali

Applichiamo LR(1) a G∉SLR(1)

I0:Closure({S'::=.S}) ={S'::= .S,$
S::= .Au,$, S::= .av,$, S::= .Bv,$,
A::= .a,u, B::=.xA,v}
I1:Goto(I0,S) ={S'::= S.,$}
I2:Goto(I0,A)={S::= A.u,$}
I3:Goto(I0,a) ={S::= a.v,$
A::= a.,u} nessun conflitto
I4:Goto(I0,B) ={S::= B.v,$}
I5:Goto(I0,x) ={S::= x.A,v
A::= .a,v}
I6:Goto(I2,u) ={S::= Au.,$}
I7:Goto(I3,v) ={S::= av.,$}
I8:Goto(I4,v) ={S::= Bv.,$}
I9:Goto(I5,A) ={S::= xA.,v}
I10:Goto(I5,a) ={A::= a.,v}

S::= A u | a v | B v
A::= a
B::= x A

Complessità in Spazio

LL(1) ordine N*T   (dove N nonterminali ed T terminali)

SLR   ordine N*P*T   (dove P dimensione parte destra prod)

LR(1)   ordine N*P*T*T

LALR(1)   come   SLR

Esempio

S::=Sb | caSd | c

ESEMPIO: S::=Sb | caSd | c

I0:{S'::=.S S::=.Sb S::=.caSd S::=.c}

I1=Goto(I0,S)={S'::=S. S::=S.b}

I2=Goto(I0,c)={S::=c.aSd S::=c.}

I3=Goto(I1,b)={S::=Sb.}

I4=Goto(I2,a)={S::=ca.Sd S::=.Sb S::=.caSd S::=.c}

I5=Goto(I4,S)={S::=caS.d S::=S.b}

Goto(I4,c)=I2

I6=Goto(I5,d)={S::=caSd.}

kernel

Esempio - continua

S::=Sb | caSd | c

I0:{S'::=.S,\$ S::=.Sb,\$/b   S::=.caSd,\$/b   S::=.c,\$/b}

I1=Goto(I0,S)={S'::=S.,\$ S::=S.b,\$/b}

I2=Goto(I0,c)={S::=c.aSd,\$/b   S::=c.,\$/b}

I3=Goto(I1,b)={S::=Sb.,\$/b}

I4=Goto(I2,a)={S::=ca.Sd,\$/b   S::=.Sb,d/b
S::=.caSd,d/b   S::=.c,d/b}

I5=Goto(I4,S)={S::=caS.d,\$/b   S::=S.b,d/b}

I6=Goto(I4,c)={S::=c.aSd,d/b   S::=c.,d/b}

I7=Goto(I5,d)={S::=caSd.,\$/b}=Goto(I10,b)

I8=Goto(I5,b)={S::=Sb.,d/b}=

I9=Goto(I6,a)={S::=ca.Sd,d/b   S::=.Sb,d/b
S::=.caSd,d/b   S::=.c,d/b}

I10=Goto(I9,S)={S::=caS.d,d/b   S::=S.b,d/b}

Goto(I9,c)=I6

I11=Goto(I10,d)={S::=caSd.,d/b}

Da *SLR(1) & LR(1)* A LALR(1)

Uno stato SLR

$\{t_1\ t_2\ \ldots\ t_k\}$

e' duplicato in n stati LR(1)

$\{t_1,U_{11}\quad t_2,U_{21}\ \ldots\quad t_k,U_{k1}\}$

......

$\{t_1,U_{1n}\quad t_2,U_{2n}\ldots\ t_k,U_{kn}\}$

LALR(1)

mettiamo insieme i duplicati

$\{t_1,U_{11}\cup\ldots\cup U_{1n}\quad t_2,U_{21}\cup\ldots\cup U_{2n}\ \ldots\ t_k,U_{k1}\cup\ldots\cup U_{kn}\ \}$

Collezione stati LALR: $Coll_{LALR}$

Sia $\Leftrightarrow$ la relazione di equivalenza su Coll(1) cosi' definita:

$$I_j \Leftrightarrow I_k \textit{ se e solo se } I_j{\downarrow}0 = I_k{\downarrow}0$$

dove $I{\downarrow}0 = \{A::=\alpha.\beta \mid A::= \alpha.\beta,U \in I\}$

$Coll_{LALR}$ = Coll(1)/$\Leftrightarrow$ = $\{I/\Leftrightarrow \mid I \in Coll(1)\}$
(partizione di $\Leftrightarrow$ su Coll(1))

dove $I/\Leftrightarrow = h \in \{J \in Coll(1) \mid J \Leftrightarrow I\}$

$\Downarrow$

$(\forall\ I_j,I_k \in Coll(1)),(\forall\ a \in N \cup T)$,

I_j=Goto(I_k,a) *se e solo se* $I_j/\Leftrightarrow$ =Goto($I_k/\Leftrightarrow$,a)

La funzione Goto del riconoscitore LALR

Cosi facendo ritroviamo i conflitti di SLR ?

shift/reduce

$A ::= \alpha.,U \in Ij$ and $B ::= \beta.a\delta,v \in Ij$ and $a \in U$

$U = U_1 \cup \ldots \cup U_n$

$\Downarrow$

conflitto presente anche in LR(1)

reduce/reduce

LR(1) potrebbe non averlo

LALR(1)

proprietà1: (No shift-reduce)

$(\forall\ I \in Coll_{LALR})\ (\forall\ A::=\alpha.a\gamma,V\ ,\ B::=\beta.,U\ \in I)$

$a \notin U$

proprietà2: (No reduce-reduce)

$(\forall\ I \in Coll_{LALR}),\ (\forall\ A::=\alpha.,V\ ,\ B::=\beta.,U\ \in I)$

$V \cap U = \{\}$

Th. Data una grammatica G:

- G∈LALR(1) se e solo se entranbe prop.1 e prop.2
- G ha parser a riduzione 1-simbolo LALR
  se e solo se entranbe prop.1 e prop.2

Bottom-up: Conclusioni

1. Cosa significa l'inclusione stretta SLR(1) $\subset$ LALR(1) $\subset$ LR(1)?

2. Per ogni linguaggio **T** e' decidibile esistenza G tale che:
L(G) = **T** ed in piu' G $\in$LALR(1) [oppure G $\in$LR(1)] ?

3. Perchè cercando un parser 1-simbolo bottom-up per G cerchiamo anzitutto di provare G $\in$SLR(1) ?

Bottom-up: Conclusioni (continua)

4. Perchè se una grammatica $G \notin LALR(1)$ a causa della prop.2 potrebbe nondimeno essere $G \in LR$?

5. Perchè $SLR(1) \subset LALR(1)$?
   - Sapreste mostrare che ogni $G \in SLR(1)$ è anche $G \in LALR(1)$?
   - Conoscete una grammatica G che provi che tale inclusione è stretta ?

6. Se una grammatica G è ambigua quale proprietà sarà violata: prop.1 o prop.2 ?
   Perchè proprio quella ?